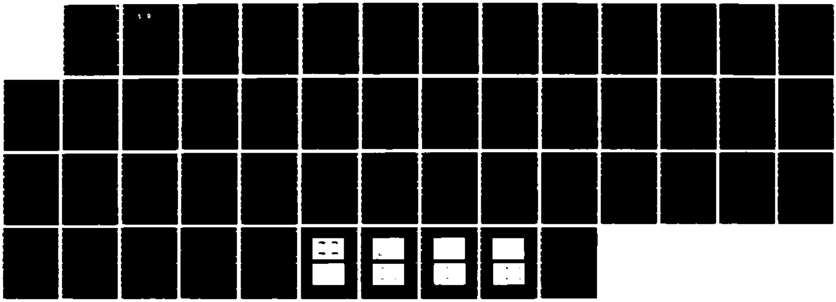
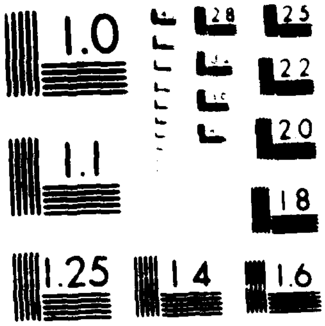
AD-A179 904 IMAGE MODELING: A MATHEMATICAL FRAMEWORK FOR SEGMENTATION AND OBJECT DETE..(U) STATISTICAL CONSULTING ASSOCIATES INC PROVIDENCE RI 1/1

UNCLASSIFIED D E MCCLURE ET AL. 20 MAR 87 F/G 20/6 NL





MICROCOPY RESOLUTION TEST CHART

NATIONAL BUREAU OF STANDARDS-1963-A

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE

(2)

REPORT DOCUMENTATION PA

AD-A179 904

1a REPORT SECURITY CLASSIFICATION Unclassified	1b RESTRICTIVE MA
2a SECURITY CLASSIFICATION AUTHORITY	3 DISTRIBUTION/A Approved for public release; distribution unlimited.
2b DECLASSIFICATION / DOWNGRADING SCHEDULE	
4 PERFORMING ORGANIZATION REPORT NUMBER(S)	5 MONITORING ORGANIZATION REPORT NUMBER(S) ARO 20746.1-MA-S
6a NAME OF PERFORMING ORGANIZATION Statistical Consulting Associates, Inc. — 6b OFFICE SYMBOL (If applicable)	7a NAME OF MONITORING ORGANIZATION U. S. Army Research Office
6c ADDRESS (City, State, and ZIP Code) [illegible] [illegible] 02146-0426	7b ADDRESS (City, State, and ZIP Code) P. O. Box 12211 Research Triangle Park, NC 27709-2211
8a NAME OF FUNDING / SPONSORING ORGANIZATION U. S. Army Research Office — 8b OFFICE SYMBOL (If applicable)	9 PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER DAAG29-83-C-0031
8c ADDRESS (City, State, and ZIP Code) P. O. Box 12211 Research Triangle Park, NC 27709-2211	10 SOURCE OF FUNDING NUMBERS

PROGRAM ELEMENT NO	PROJECT NO	TASK NO.	WORK UNIT ACCESSION NO

11 TITLE (Include Security Classification)
[illegible] : A MATHEMATICAL FRAMEWORK FOR SEGMENTATION AND OBJECT DETECTION

12 PERSONAL AUTHOR(S)
McClure, Donald E.; Geman, Stuart; Geman, Donald; Grenander, Ulf

13a TYPE OF REPORT Final Report	13b TIME COVERED FROM 10/1/83 TO 4/30/87	14 DATE OF REPORT (Year, Month, Day) 20 March 1987	15 PAGE COUNT 48

16 SUPPLEMENTARY NOTATION The view, opinions and/or findings contained in this report are those of the author(s) and should not be construed as an official Department of the Army position, policy, or decision, unless so designated by other documentation.

17 COSATI CODES			18 SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	Image modeling, image processing, image analysis; statistical methods, Bayesian methods; Mathematical image models; FLIR, Infrared images; segmentation, object detection, edge detection

19 ABSTRACT (Continue on reverse if necessary and identify by block number)

Statistical rules for segmenting scenes and for detecting the presence of distinguished objects in digital images can be based on classical principles of statistical inference if appropriate mathematical models are developed for observable pictures. The main goal of this project was to devise and analyze alternative image models for digitized FLIR images. The work has been done in close cooperation with the Advanced Modeling Team of the U.S. Army Night Vision and Electro-Optics Laboratory, Ft. Belvoir, Virginia. This report summarizes the hierarchical Markov Random Field models and their application to restoration and segmentation of FLIR images.

20 DISTRIBUTION/AVAILABILITY OF ABSTRACT ☐ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT ☐ DTIC USERS	21 ABSTRACT SECURITY CLASSIFICATION Unclassified	
22a NAME OF RESPONSIBLE INDIVIDUAL	22b TELEPHONE (Include Area Code)	22c OFFICE SYMBOL

83 APR edition may be used until exhausted
All other editions are obsolete

SECURITY CLASSIFICATION OF THIS PAGE
UNCLASSIFIED

ARO 20746.1-MA-S

Image Modeling:
A Mathematical Framework for Segmentation and Object Detection

FINAL TECHNICAL REPORT

Donald Geman
Stuart Geman
Ulf Grenander
Donald E. McClure

20 March 1987

U.S. Army Research Office

Contract No. DAAG29-83-C-0031

Statistical Consulting Associates, Inc.
P.O. Box 2476 (148 Waterman Street)
Providence, Rhode Island 02906-0476

**APPROVED FOR PUBLIC RELEASE;
DISTRIBUTION UNLIMITED.**

VIEWS, OPINIONS, AND/OR FINDINGS CONTAINED IN THIS REPORT ARE THOSE OF THE AUTHOR(S) AND SHOULD NOT BE CONSTRUED AS AN OFFICIAL DEPARTMENT OF THE ARMY POSITION, POLICY, OR DECISION UNLESS SO DESIGNATED BY OTHER OFFICIAL DOCUMENTATION

Abstract

Decision rules for segmenting scenes and for detecting the presence of distinguished objects in digital images can be based on classical principles of statistical inference if appropriate mathematical models are developed for observable pictures. The main goal of this research was to devise and analyze alternative image models for digitized FLIR images. The work has been done in close cooperation with the Advanced Modeling Team of the U.S. Army Night Vision and Electro-Optics Laboratory, Ft. Belvoir, Virginia. This report concentrates on hierarchical Markov Random Field models and their application to restoration and segmentation of FLIR images.

Contents

1 INTRODUCTION.

Our primary goal has been to construct a mathematical foundation for the rational choice of decision functions for image analysis. This has included structured models for the background against which certain objects, such as tanks, trucks, or armored personnel carriers, appear. The backgrounds are "complex" in that their composition is highly variable and cannot be known in advance. The objects are "simple" in that they can be characterized by a small number of parameters. While the emphasis has been placed on the logical and mathematical foundations,considerable effort has been given to the construction of algorithms. It is important to keep the algorithmic issues in mind so that we arrive at decision procedures that work and that can be computed with reasonable resources.

This report focuses on a strategy for image modeling that has been developed for a number of practical settings. Here we develop it for the analysis of FLIR images. Indeed, this project—while it is immediately concerned with problems suggested by the U.S. Army Night Vision and Electro-Optics Laboratory—has had a tremendous impact on the development of a general Bayesian methodology for automatic analysis of digital images. Today that methodology is successfully addressing practical problems in medical imaging (computed tomography, ultrasound), remote sensing (interpretation of SAR images), automatic inspection (analysis of textured optical images of silicon wafers), and image understanding (optical character recognition, boundary finding, segmentation).

In the interest of presenting a self-contained and coherent report on mathematical models for FLIR images, we shall concentrate this paper on the general Bayesian model and its adaptation to FLIR imagery. Our interactions with the Advanced Modeling Team at NV&EOL have had many other facets, including frequent on-site working sessions, supervision of the development of computer algorithms, direction for the formation of a data base of features of FLIR images, statistical analyses, and assistance with providing information on other mathematical modeling efforts. These interactions are all directly related to the overall project on image modeling, and are documented elsewhere. In particular, the internal working memoranda listed in Appendix A provide additional details on both theoretical and practical aspects of the effort.

Section 2 of this paper gives an overview and basic examples of the Bayesian modeling strategy. It covers the range of issues from specification of the probabilistic framework to the design of computational algorithms.

Section 3 describes the adaptation of the general Bayesian paradigm to digitized FLIR images. Here we describe a specific heierarchical probabilistic model which is useful for FLIR image restoration and segmentation.

Section 4 presents a FORTRAN implementation of the image restoration algorithm.

Program listings are included.

Section 5 briefly describes the application of the restoration algorithm to eight examples of FLIR images provided to us by NV&EOL.

Finally, two appendices include, respectively, (i) a list of internal working papers developed durinp the project and previously shared with the Advanced Modeling Team at NV&EOL and (ii) pictures illustrating the examples cited in Section 5.

We gratefully acknowedge the contributions made to this research effort by Frank Shields and Vince Mirelli of the Advanced Modeling Team at NV&EOL. The discussions of the fundamental mathematical issues with Dr. Mirelli have provided a tremendous stimulus for focusing our efforts on meaningful ways of bringing mathematics to bear on challenging practical problems.

2 BAYESIAN PARADIGM.

In real scenes, neighboring pixels typically have similar intensities, boundaries are usually smooth and often straight, textures, although sometimes random locally, define spatially homogeneous regions, and objects, such as grass, tree trunks, branches and leaves, have preferred relations and orientations. Our approach to picture processing is to articulate such regularities mathematically, and then to exploit them in a statistical framework to make inferences. The regularities are rarely deterministic; instead, they describe correlations and likelihoods. This leads us to the Bayesian formulation, in which prior expectations are formally represented by a probability distribution. Thus we design a distribution (a "prior") on relevant scene attributes to capture the tendencies and constraints that characterize the scenes of interest. Picture processing is then guided by this prior distribution, which, if properly conceived, enormously limits the plausible restorations and interpretations.

The approach involves five steps, which we shall briefly review here (see [4] and [9] for more details). This will define the general framework, and then, in the following sections, we will concentrate on the analysis of samples of FLIR images, as an illustrative application.

2.1 Image Models.

These are probability distributions on relevant image attributes. Both for reasons of mathematical and computational convenience, we use *Markov random fields* (MRF) as prior probability distributions. Let us suppose that we index all of the relevant attributes by the index set S. S is application specific. It typically includes indices for each of the pixels (about 512x512 in the usual video digitization) and may have other indices for such attributes as boundary elements, texture labels, object labels and so-on. Associated with each "site" $s \in S$ is a real-valued random variable X_s, representing the state of the corresponding attribute. Thus X_s may be the measured intensity at pixel s (typically, $X_s \in \{0, ...255\}$), or simply 1 or 0 as a boundary element at location s is present or absent.

The kind of knowledge we represent by the prior distribution is usually "local," which is to say that we articulate regularities in terms of small local collections of variables. In the end, this leads to a distribution on $X = \{X_s\}_{s \in S}$ with a more or less "local neighborhood structure" (again, we refer to [4] and [9] for details). Specifically, our priors are Markov random fields: there exists a (symmetric) *neighborhood relation* $G = \{G_s\}_{s \in S}$, wherein $G_s \subseteq S$ is the set of neighbors of s, such that

$$\Pi(X_s = x_s | X_r = x_r, r \in S, r \neq s) = \Pi(X_s = x_s | X_r = x_r, r \in G_s)$$

$\Pi(a|b)$ is conditional probability, and, by convention, $s \notin G_s$. G symmetric means $s \in G_r \Leftrightarrow r \in G_s$. (Here, we assume that the range of the random vector X is discrete; there are obvious modifications for the continuous or mixed case.)

It is well-known, and very convenient, that a distribution Π defines a MRF on S with neighborhood relation G if and only if it is Gibbs with respect to the same graph, (S, G). The latter means that Π has the representation

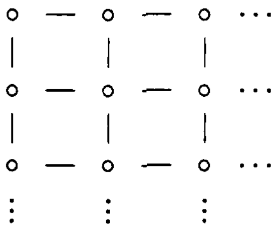
$$\Pi(x) = \frac{1}{z} e^{-U(x)} \tag{2.1}$$

where

$$U(x) = \sum_{c \in C} V_c(x) \tag{2.2}$$

C is the collection of all cliques in (S, G) (collections of sites such that every two sites are neighbors), and $V_c(x)$ is a function depending only on $\{x_s\}_{s \in c}$. U is known as the "energy," and has the intuitive property that the low energy states are the more likely states under Π. The normalizing constant, z, is known as the "partition function". The Gibbs distribution arises in statistical mechanics as the equilibrium distribution of a system with energy function U.

As a simple example (too simple to be of much use for real pictures) suppose the pixel intensities are known, a priori, to be one of two levels, minus one ("black") or plus one ("white"). Let S be the $N \times N$ square lattice, and let G be the neighborhood system that corresponds to nearest horizontal and vertical neighbors:



For picture processing, think of N as typically 512. Suppose that the only relevant regularity is that neighboring pixels tend to have the same intensities. An "energy" consistent with this regularity is the "Ising" potential:

$$U(x) = -\beta \sum_{(s,t)} x_s x_t \qquad \beta > 0$$

where $\sum_{(s,t)}$ means summation over all neighboring pairs $s, t \in S$. The minimum of U is achieved when $x_s = x_t \quad \forall s, t \in S$. Under (2.1), the likely pictures are therefore the ones that respect our prior expectations; they segment into regions of constant intensities. The larger β, the larger the typical region. Later we will discuss the issue of *estimating* model parameters such as β. (With energy (2.2), Π in (2.1) is called the Ising model. It models the equilibrium distribution of the spin states of the atoms in a ferromagnet. These spins tend to "line up," and hence the favored configurations contain connected regions of constant spins.)

One very good reason for using MRF priors is their Gibbs representations. Gibbs distributions are characterized by their energy functions, and these are more convenient and intuitive for modelling than working directly with probabilities. See, for example, [3], [4], [5], [9], and [13] for many more examples, and Section 3 below for a more complex and useful MRF model.

2.2 Degradation Model.

The image model is a distribution $\Pi(\cdot)$ on the vector of image attributes $X = \{X_s\}_{s \in S}$. *By design*, the components of this vector contain all of the relevant information for the image processing task at hand. Hence, the goal is to estimate X. This estimation will be based upon partial or corrupted observations, and based upon the prior information. In emission tomography, X represents the spatial distribution of isotope in a target region of the body. What is actually observed is a collection of photon counts whose probability law is Poisson, with a mean function that is an attenuated radon transform of X. In the texture labelling problem, X is the pixel intensity array and a corresponding array of texture labels. Each label gives the texture type of the associated pixel. The observation is only partial: we observe the pixels, which are just the digitized picture, but not the labels. The purpose is then to estimate the labels from the picture. In a generic model for FLIR images described in Section 3, X is a hierarchical model built from the pixel intensity array and from a superimposed array of unobservable edge elements. Again, the observation is only partial: we observe the pixels, degraded as they are by atmospheric effects and the sensor, but not the edge elements that are combined to form boundaries between objects and background. A purpose of image segmentation is to estimate the boundaries from the observed picture.

The observations are related to the image process (X) by a *degradation model*. This models the relation between X and the *observation process*, say $Y = \{Y_s\}_{s \in T}$. For texture analysis, we will define $X = (X^P, X^L)$, where X^P is the usual grey-level pixel intensity process, and X^L is an associated array of texture labels. The observed picture is just X^P, and hence $Y = X^P$: the degradation is a projection. More typically, the degradation involves a random component, as in the tomography setting where the observations are Poisson variables whose means are related to the image process X. A more simple, and widely studied (if unrealistic) example is additive "white" noise. Let $X = \{X_s\}_{s \in S}$ be just the basic pixel process. In this case, $T = S$, and for each $s \in S$ we observe

$$Y_s = X_s + \eta_s$$

where, for example, $\{\eta_s\}_{s \in S}$ is Gaussian with independent components, having means 0 and variances σ^2.

Formally, the degradation model is a conditional probability distribution, or density, for Y given X: $\Pi(y|x)$. If the degradation is just added "white noise," as in the above example, then

$$\Pi(y|x) = \left(\frac{1}{2\pi\sigma^2}\right)^{\frac{|S|}{2}} exp\{-\frac{1}{2\sigma^2}\sum_{s \in S}(y_s - x_s)^2\}$$

For labelling textures, the degradation is deterministic: $\Pi(y|x)$ is concentrated on $y = x^P$, where $x = (x^P, x^L)$ has both pixel and label components.

2.3 Posterior Distribution.

This is the conditional distribution on the image process X given the observation process Y. This "posterior" or *"a posteriori"* distribution contains the information relevant to the image restoration or image analysis task. Given an observation $Y = y$, and assuming the image model ($\Pi(x)$) and degradation model ($\Pi(y|x)$), the posterior distribution reveals the likely and unlikely states of the "true" (unobserved) image X. Having constructed X to contain all relevant image attributes, such as locations of boundaries, labels of objects or textures, and so-on, the posterior distribution comes to play the fundamental role in our approach to image processing.

The posterior distribution is easily derived from "Bayes' rule"

$$\Pi(x|y) = \frac{\Pi(y|x)\Pi(x)}{\Pi(y)}$$

The denominator, $\Pi(y)$, is difficult to evaluate. It derives from the prior and degradation models by integration: $\Pi(y) = \int_x \Pi(y|x)\Pi(dx)$, but the formula is computationaly intractable. Happily, our analysis of the posterior distribution will require only *ratios*, not absolute probabilities. Since y is fixed by observation, $\frac{1}{\Pi(y)}$ is a constant that can be ignored (see paragraph below on "computing").

As an example we consider the simple "Ising model" prior, with observations corrupted by additive white noise. Then

$$\Pi(x) = \frac{1}{z} exp\{-\beta \sum_{(s,t)} x_s x_t\}$$

and

$$\Pi(y|x) = \left(\frac{1}{2\pi\sigma^2}\right)^{\frac{|S|}{2}} exp\{-\frac{1}{2\sigma^2}\sum_{s\in S}(y_s - x_s)^2\}$$

The posterior distribution is then

$$\Pi(x|y) = \frac{1}{z_p}exp\{-\beta\sum_{(s,t)} x_s x_t - \frac{1}{2\sigma^2}\sum_{s\in S}(y_s - x_s)^2\}$$

We denote by z_p the normalizing constant for the posterior distribution. Of course, it depends upon y, but the latter is fixed. Notice that the posterior distribution is again a MRF. In the case of additive white noise, the neighborhood system of the posterior distribution is that of the prior, and hence local. For a wide class of useful degradation models, including combinations of blur, added or multiplicative "colored noise," and a variety of nonlinear transformations, the posterior distribution is a MRF with a more or less local graph structure. This is convenient for our computational schemes, as we shall see shortly. We should note, however, that exceptions occur. In tomography, for example, the posterior distribution is associated with a highly non-local graph. This situation incurs a high computational cost (see [5] for more details).

2.4 MAP Estimate.

In our framework, image processing amounts to choosing a particular image x, given an observation $Y = y$. A sensible, and suitably-defined optimal, choice is the "maximum a posteriori," or "MAP" estimate:

$$\text{choose } x \text{ to maximize } \Pi(x|y)$$

The MAP estimate chooses the most likely x, given the observation. In most applications, our goal is to identify the MAP estimate, or a suitable approximation. However, in some settings other estimators are more appropriate. We have found, for example, that the posterior mean ($\int x\Pi(dx|y)$) is more effective for tomography, at least in our experiments. Here, we concentrate on MAP estimation.

In most applications we can not hope to identify the true maximum a posteriori image vector x. To appreciate the computational difficulty, consider again the Ising model with added white noise:

$$\Pi(x|y) = \frac{1}{z_p}exp\{-\beta\sum_{(s,t)} x_s x_t - \frac{1}{2\sigma^2}\sum_{s\in S}(y_s - x_s)^2\} \tag{2.3}$$

This is to be maximized over all possible vectors $x = \{x_s\}_{s \in S} \in \{-1,1\}^{|S|}$. With $S \sim 10^5$, brute force approaches are intractable; instead, we will employ a Monte Carlo algorithm which gives adequate approximations.

Maximizing (2.3) amounts to minimizing

$$U_p(x) = -\beta \sum_{(s,t)} x_s x_t - \frac{1}{2\sigma^2} \sum_{s \in S} (y_s - x_s)^2$$

which might be thought of as the "posterior energy". (As with z_p, the fixed observation y is suppressed in the notation $U_p(x)$.) More generally, we write the posterior distribution as

$$\frac{1}{z_p} exp\{-U_p(x)\} \tag{2.4}$$

and characterize the MAP estimator as the solution to the problem

choose x to minimize $U_p(x)$

The utility of this point of view is that it suggests a further analogy to statistical mechanics, and a computation scheme for approximating the MAP estimate, which we shall now describe.

2.5 Computing.

Pretend that (2.4) is the equilibrium Gibbs distribution of a real system. Recall that MAP estimation amounts to finding a minimal energy state. For many physical systems the low energy states are the most ordered, and these often have desirable properties. The state of silicon suitable for wafer manufacturing, for example, is a low energy state. Physical chemists achieve low energy states by heating and then slowly cooling a substance. This procedure is called *annealing*. Cerný [1] and Kirkpatrick [12] suggest searching for good minimizers of $U(\cdot)$ by *simulating* the dynamics of annealing, with U playing the role of energy for an (imagined) physical system. In our image processing experiments, we often use "simulated annealing" to find an approximation to the MAP estimator.

Dynamics are simulated by producing a Markov chain, $X(1), X(2), \ldots$ with transition probabilities chosen so that the equilibrium distribution is the posterior (Gibbs) distribution (2.4). One way to do this is with the "Metropolis algorithm" [14]. More convenient for image processing is a variation we call *stochastic relaxation*. The full story can be found in [4] and [9]. Briefly, in stochastic relaxation we choose a sequence of sites

$s(1), s(2), \ldots \in S$ such that each site in S is "visited" infinitely often. If $X(t) = x$, say, then $X_r(t+1) = x_r \ \forall r \neq s(t), r \in S$, and $X_{s(t)}(t+1)$ is a sample from

$$\Pi(X_{s(t)} = \cdot | X_r = x_r, r \neq s(t)),$$

the conditional distribution on $X_{s(t)}$ given $X_r = x_r \ \forall r \neq s(t)$. By the Markov property,

$$\Pi(X_{s(t)} = \cdot | X_r = x_r, r \neq s(t)) = \Pi(X_{s(t)} = \cdot | X_r = x_r, r \in G^p_{s(t)})$$

where $\{G^p_s\}_{s \in S}$ is the *posterior* neighborhood system, determined by the posterior energy $U_p(\cdot)$. The prior distributions that we have experimented with have mostly had local neighborhood systems, and usually the posterior neighborhood system is also more or less local as well. This means that $|G^p_{s(t)}|$ is small, and this makes it relatively easy to generate, Monte Carlo, $X(t+1)$ from $X(t)$. In fact, if Ω is the range of $X_{s(t)}$, then

$$(2.5) \qquad \Pi(X_{s(t)} = \alpha | X_r = x_r, r \in G^p_{s(t)}) = \frac{\Pi(\alpha, {}_{s(t)}x)}{\sum_{\hat{\alpha} \in \Omega} \Pi(\hat{\alpha}, {}_{s(t)}x)}$$

where

$$(\alpha, {}_{s(t)}x)_r = \begin{cases} \alpha & \text{if } r = s(t) \\ x_r & \text{if } r \neq s(t) \end{cases}$$

Notice that (fortunately!) there is no need to compute the posterior partition function z_p. Also, the expression on the right hand side of (2.5) involves only those potential terms associated with cliques containing $s(t)$, since all other terms are the same in the numerator and the denominator.

To simulate annealing, we introduce an artificial "temperature" into the posterior distribution:

$$\Pi_T(x) = \frac{exp\{\frac{-U_p(x)}{T}\}}{Z_p(T)}$$

As $T \to 0, \Pi_T(\cdot)$ concentrates on low energy states of U_p. To actually find these states, we run the stochastic relaxation algorithm while slowly lowering the temperature. Thus $T = T(t)$, and $T(t) \downarrow 0$. $\Pi_{T(t)}(\cdot)$ replaces $\Pi(\cdot)$ in computing the transition $X(t) \to X(t+1)$. In [4] we showed that, under suitable hypotheses on the sequence of site visits, $s(1), s(2), \ldots$:

> If $T(t) > \frac{c}{1+log(1+t)}$, $T(t) \downarrow 0$, then for all c sufficiently large $X(t)$ converges weakly to the distribution concentrating uniformly on $\{x : U(x) = min_y U(y)\}$.

More recently, our theorem has been improved upon by many authors. In particular, the smallest constant c which guarantees convergence of the annealing algorithm to a global minimum can be specified in terms of the energy function U_p (see [6] and [10]). Also, see Gidas [7] for some ideas about faster annealing via "renormalization group" methods.

In the experiments with FLIR images to be described here, MAP estimates are approximated by using the annealing algorithm. This involves Monte Carlo computer-generation of the sequence $X(1), X(2), \ldots$, terminating when the state ceases to change substantially.

3 A GENERIC OBJECT/BACKGROUND MODEL.

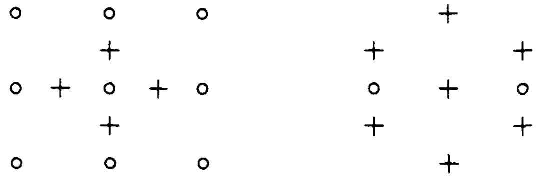
The general modeling strategy described in Section 2 has been implemented for FLIR images with immediate objectives of image restoration (i) to "smooth" and enhance homogeneous subregions corresponding, for example, to an object or to a large component of an object of interest, and (ii) to highlight boundaries between separate homogeneous subregions as a precurser to object detection and recognition. We have designed and implemented a two-level hierarchical MRF model combining the directly observable pixel process and a linked unobservable binary process indicating the presence or absence of edge elements. Models like the one described here were suggested and illustrated in [2].

3.1 Scene Model.

The image process X comprises the pixel process X^P and the edge process X^E, $X = \{X^P, X^E\}$. As usual, the pixel sites form an $R \times C$ array of points (R rows and C columns) in a square lattice arrangement. We denote this $R \times C$ array by S^P. The sites for the edge process, collectively denoted S^E, also form a regular graph structure, envisioned as fitting between the sites in S^P. Let u, v denote pixel sites in the square lattice S^P, For each pair u, v of horizontally or vertically adjacent pixels, there exists an "edge site" denoted $< u,v >$ in S^E. The edge site $s =< u,v >$ corresponds to the location of possible edge or boundary element between pixels u and v. The edge variables are binary, with $X^E_{<u,v>}$ equalling 1 or 0 to indicate the presence or absence of an edge element at $< u,v >$. The process X^E consists of $R(C-1) + C(R-1)$ variables $X^E_{<u,v>}$.

The totality of edge and pixel sites is denoted by S. (The generic point s may refer to a pixel or to an edge site $< u,v >$.) The neighborhood system $G = \{G_s, s \in S\}$ governs the Markovian dependence structure of $X = \{X^P, X^E\}$. The size of the neighborhood determines the range of interactions. We restrict our design of the process to "small" or "local" neighborhood sets G_s, to keep the mathematical models as simple as possible and to assure feasibility of computational procedures.

We adopt the following neighborhood system. Each pixel site has eight pixel neighbors, the nearest ones, and four edge neighbors. Each edge site $< u,v >$ has six edge neighbors—corresponding to possible continuations of a boundary from $< u,v >$—and the two pixel neighbors u and v. Sites near the boundary of the lattice have fewer neighbors. The canonical pixel neighborhood G_s and edge neighborhood $G_{<s,t>}$ are depicted in the figure below, where circles represent pixels and pluses represent edge sites. (We believe this edge graph originated in [11].)



To illustrate the functional form of the models, suppose first that we are only interested in modeling "smoothness" or "regularity" in the intensity array X^P, i.e., the tendency of nearby pixels to have similar intensities. Then a suitable model might be $X = X^P$ with

$$\Pi(X = x) = Z^{-1} \exp\{\theta \sum_{(s,t)} C_{(s,t)} \phi(x_s - x_t)\}.$$

where the sum extends over all neighboring pairs (s,t) of pixels. (Thus each interior pixel is included in eight terms in the summation.) Here $\phi = \phi(\delta)$ is an even function, decreasing for $\delta > 0$; θ is a parameter which corresponds to "inverse temperature" and it governs the degree of regularity. It is distinct from the "artificial temperature" T introduced for the annealing algorithm (Section 2.5). The coefficient $C_{(s,t)}$ is introduced to allow different weighting of pixel pairs oriented in different directions. We commonly fix $C_{(s,t)} = 1$ for the horizontal and vertical pairs and $C_{(s,t)} = 1/\sqrt{2}$ for diagonally adjacent pairs. A renormalization argument shows that this weighting is "asymptotically correct" in order for the discrete digital images X^P to approximate rotationally invariant (isotropic) images on a continuous background [8]. The weights also permit accurate modeling of anisotropic FLIR images.

A flexible and well-tested choice for ϕ is of the form

$$\phi(\delta) = \left(1 + \left|\frac{\delta}{B}\right|^2\right)^{-1} \tag{3.1}$$

where B is a parameter depending on the dynamic range of the image. An attractive feature of this ϕ-function—in contrast to one that decreases without bound—is that it *does not* attach ever increasing penalties to larger differences δ, and thus it will allow sharp gradients in intensity across region boundaries. A choice such as $\phi(\delta) = -\delta^2$ would *a priori* inhibit, indeed prohibit, adjacent, internally homogeneous subregions with highly separated intensities.

With the inclusion of the edge process X^E we incorporate our expectations about both the interactions between intensities and edges—i.e., where the edges belong—and about clusters of nearby edges. We are not exactly modeling entire boundaries with this two-level model, but rather *segments* of boundaries; except in the simplest imagery and with larger neighborhoods, it is essentially impossible to distinguish actual boundary segments from intensity gradients due to lighting, texture, etc.

For the pixel-edge process, the complete energy function $U = U(X^P, X^E)$ is decomposed into two components:

$$U(X^P, X^E) = U^1(X^P, X^E) + U^2(X^E).$$

We construct U^1 so that the most likely configurations will have $X^E_{<s,t>} = 1$ (respectively 0) when the intensity difference $|x^P_s - x^P_t|$ between neighboring pixels is large (resp. small). Put differently, we want to "break" the bond between pixels s and t when their values are "far" apart. Thus we choose

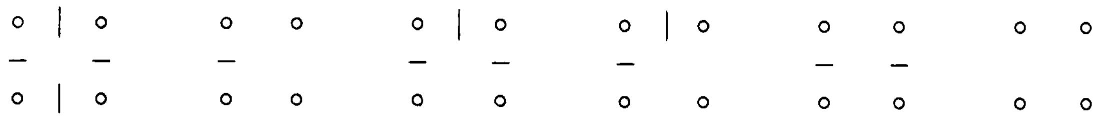
$$(3.2) \qquad U^1(x^P, x^E) = -\sum_{(s,t)} \theta_1 C_{(s,t)}(\phi(x^P_s - x^P_t) - \theta_2) \times (1 - I_{(s,t)}(X^E)) \qquad .$$

where $\theta_1 > \theta_2 > 0$. The value of δ for which $\theta_1 C_{(s,t)}\phi(\delta) = \theta_2$ represents an intensity difference for which we have "no preference" in regard to the on-off state of an edge; such interpretations of the model parameters are helpful when one is setting or estimating values of the parameters. Finally, in equation (3.2), $I_{(s,t)}(X^E) = 1$ when the X^E process "breaks" the bond between pixels s and t, and $I_{(s,t)}(X^E) = 0$ otherwise. In particular, if s and t are horizontal or vertical neighbors, then $I_{(s,t)}(X^E) = X^E_{<s,t>}$ and if s and t are diagonal neighbors, then $I_{(s,t)}(X^E)$ is a Boolean function of four edge elements between s and t requiring, for its value to be 1, that at least two of the edge elements are "on" and that they connect to form a segment separating s from t.

The remaining component U^2 of the total energy function governs the organization of nearby edges. We define

$$U^2(x^E) = -\theta_3 \sum_D V_D(x^E)$$

where $\theta_3 > 0$ and where the sum extends over all subsets D of four neighboring edge sites—the maximal "cliques" in the edge graph. The clique function V_D assigns weights in accordance with our expectations about edge behavior. More specifically, there are six possible clique states, up to rotational equivalence:



Here the bars indicate that the edge variable at the indicated site is "on". Let $V_D = \xi_i$, for $i = 1, \ldots, 6$, denote the weights assigned to the six configurations above. If we assume that most pixels are not next to boundaries, that edges should continue, and that boundary congestion is unlikely, then we might choose $\xi_1 \leq \xi_2 \leq \xi_3 \leq \xi_4 \leq \xi_5 \leq \xi_6$. A specific image-dependent choice is made in the experiment described in Section 5.

One final point about the scene model: it is useful to write the total energy, up to an additive constant, as

$$(3.3) \quad -U(x) = \theta_1 \sum_{(s,t)} C_{(s,t)}\phi(x_s^P - x_t^P)(1 - I_{(s,t)}(x^E)) + \theta_2 \sum_{(s,t)} I_{(s,t)}(x^E) + \theta_3 \sum_{D} V_D(x^E)$$

For inferential purposes, this shows that our model is an *exponential family* in $\theta = (\theta_1, \theta_2, \theta_3)$. In addition, the form in (3.3) is useful for parameter interpretation; for instance, it becomes clear that θ_2 is a "reward" for edges.

3.2 Degradations.

The Gibbs distribution determined by the energy function U in equation (3.3) models the ideal scenes. There are several types of degradations that corrupt an ideal scene before it is observed. Most of these effects are well understood and can be modeled accurately in terms of the physical processes that underlie them. In the end, the first approximation of the degraded observed image Y will reduce to the pixel process X^P plus additive noise. The approximation is a gross simplification, even if it is reasonably effective as a basis for restoration algorithms. Ongoing research is exploring the use of more accurate degradation models which incorporate degradations modeled by convolutions; as we describe below, these latter degradations include atmospheric absorption and scattering, diffraction from geometric optics, and blurring from signal averaging and sampling by the IR sensor.

Two useful references for understanding degradations of IR images are the NV&EOL Technical Report [16] by J.A. Ratches et. al. and the NV&EOL internal working paper [15] prepared for our project by V. Mirelli. Some of the basic physics of IR radiation and detection is described in [17].

The primary sources of IR image degradation are:

- The actual thermal radiation from the ideal scene is random and additive to X^P. The random component has mean value 0 and has signal-dependent variance proportional to X^P. The exact distribution of the emitted radiation is well-modeled by a Poisson process and a Gaussian approximation is justified by convergence of the Poisson Law to the Normal.

- During atmospheric transmission of the radiation, there is absorption—dependent on air temperature and relative humidity—and scattering—dependent on visibility.

The scattering is normally modeled by Beer's Law [16]. The effects of absorption and scattering enter the mathematical model in the form of a convolution of the signal with a kernel depending on atmospheric parameters and range-to-target.

- At the sensor, the first degradation stems from optical diffraction. The geometrical optical effect is modeled by a convolution of the signal with a kernel depending on parameters of the optical system (lens diameter, focal lengths) and on the wavelength of the electromagnetic radiation.
- Black-body radiation from the positive temperature of the detector corrupts the image incident at the detector. This effect enters the model as additive "noise" on top of the signal.
- The electromagnetic energy in the IR radiation is converted to an electrical response by the sensor. The response is a random process subordinated on the input. This can be represented mathematically as signal-dependent additive noise, again with a Poisson distribution, where both the conditional mean and variance of the response (given the input) equals the input.
- The electrical response is digitized through a combination of averaging and sampling. Conceptually, a scanning detector returns a continuous response which is averaged in the direction of the scan and which is discretely sampled in the direction orthogonal to the scan. The combination of averaging and sampling implies that the observed process *will not* be isotropic. Digitization is described mathematically through a convolution of the continuous signal with a singular kernel.
- Finally, electronic noise may enter at the last stage of actually observing the digitized signal. The noise enters as an additive effect, independent of the signal.

4 IMPLEMENTATION OF THE RESTORATION ALGORITHM.

The following subsections give a complete listing of a standard FORTRAN77 program that implements stochastic relaxation, with optional annealing, for the model described in Section 3. The subroutine that computes the dependence of the total energy on the edge process (SUBROUTINE DEE) actually implements a model that is slightly more general than equation (3.3). It incorporates a parameter (CE2C) which inhibits the occurrence of nearby parallel edges. The model of Section 3 is implemented by this program when CE2C=0.

This program has been delivered to the Advanced Modeling Team at NV&EOL and has been used there for experiments with restoration of FLIR images. The presumptions about formats of input and output files are best documented by the input and output subroutines READIN and WRITEO, which are listed below. Experiments with the use of this program are described in Section 5.

4.1 Main Program RESTOR.

The main program guides input, output and stochastic relaxation of the pixel and edge processes.

```
      PROGRAM RESTOR
C SET UP DATA STRUCTURES
      INCLUDE (COMMON)
C TYPES
      INTEGER NIT
C GET INPUT
      CALL READIN
C ITERATE
      DO 10 NIT=NSTART,NSTOP
         PRINT *, 'ITERATION ', NIT
            IF (NIT.LE.NO) THEN
               T=TO
            ELSE
               T=TO/(1.0+LOG(FLOAT(NIT-NO)))
            ENDIF
                    PRINT *, 'TEMPERATURE ', T
            IF (IXP.EQ.1) CALL ITXP
            IF (IXE.EQ.1) CALL ITXE
10    CONTINUE
C OUTPUT RESULTS
```

```
      CALL WRITEO                                                       RES00210
      END                                                               RES00220
```

4.2 Include File COMMON.

The "include" file declares global variables, sets parameter values, and defines COMMON blocks.

CE1A is the model parameter θ_1 (equation 3.3).
CE1B is the model parameter B (equation 3.1).
CE2A is the model parameter θ_3 (equation 3.3).
CE2B is the model parameter θ_2 (equation 3.3).
CE2C is not used in model (3.3) and is set to 0.
PMIN is the minimum value of the range of the pixel process x_s^P.
PMAX is the maximum value of the range of the pixel process x_s^P.

SIGMA is the standard deviation of the additive noise corrupting the observed process$Y$.

MAXDA is the maximum number of equally spaced discrete levels used to quantize the range [PMIN,PMAX] of x_s^P.

NDA is the actual number of equally spaced discrete levels used to quantize the range [PMIN,PMAX] of x_s^P.

```
C CONSTANTS
      INTEGER NX, NY, MAXDA
      REAL DIAG
      PARAMETER (NX=64,NY=64,MAXDA=100,DIAG=.707)
C DECLARE PARAMETERS, WHICH WILL BE READ FROM UNIT 7
      REAL CE1A,CE1B,CE2A,CE2B,CE2C,PMIN,PMAX,SIGMA
C VARIABLES AND ARRAYS
      INTEGER IS, ID, IP, IXP, IXE, NO, NSTART, NSTOP,
     M     NDA
      REAL TO,T,XP(0:NX+1,0:NY+1),XE(-1:NX+2,-1:NY+2,2),XPO(NX,NY),
     M     ADSIG,SIGSQD
      DOUBLEPRECISION SEED
C COMMON GLOBAL DATA STRUCTURES
      COMMON SEED,CE1A,CE1B,CE2A,CE2B,CE2C,PMIN,PMAX,SIGMA,
     M  TO,T,XP,XE,XPO,ADSIG,SIGSQD,
     M  IS,ID,IP,IXP,IXE,NO,NSTART,NSTOP,NDA
```

4.3 Subroutine READIN.

The input routine READIN prompts the user for interactive input of program and model parameters and reads in files containing images, including the observed image and any results that may be available from previous processing by the relaxation algorithm.

```
      SUBROUTINE READIN
C SET UP DATA STRUCTURES
      INCLUDE (COMMON)
C TYPES
      INTEGER I, J, K
C EXTERNAL FUNCTIONS CALLED
      REAL GGUBFS
C READ PARAMETER VALUES FROM UNIT 7
      READ(7,*), CE1A
      READ(7,*), CE1B
      READ(7,*), CE2A
      READ(7,*), CE2B
      READ(7,*), CE2C
      READ(7,*), PMIN
      READ(7,*), PMAX
      READ(7,*), SIGMA
      CLOSE(UNIT=7)
C DETERMINE IF GOAL IS IMAGE SAMPLING
      IS=0
      WRITE(6,*), 'ENTER 1 IF A SAMPLE IMAGE IS DESIRED, 0 IF PURPOSE'
      WRITE(6,*), 'IS RESTORATION'
      READ(5,*), IS
C IF GOAL IS RESTORATION, DETERMINE IF ORIGINAL IMAGE IS RESULT OF
C A DEGRADATION
      ID=0
      IF (IS.EQ.0) THEN
         WRITE(6,*), 'ENTER 1 IF THERE IS A DEGREDATION, 0 OTHERWISE'
         READ(5,*), ID
      ENDIF
C DETERMINE IF IMAGE HAS ALREADY BEEN PARTIALLY PROCESSED
      IP=0
      WRITE(6,*), 'ENTER 1 IF PROCESSING BEGAN WITH A PREVIOUS RUN,'
      WRITE(6,*), '0 OTHERWISE'
      READ(5,*), IP
C DETERMINE WHICH LEVELS OF HIERARCHY ARE TO BE ACTIVE
      IXP=0
      IXE=0
      WRITE(6,*), 'ENTER 1 IF PIXEL PROCESS WILL BE ACTIVE, 0 OTHERWISE'
      READ(5,*), IXP
      WRITE(6,*), 'ENTER 1 IF EDGE PROCESS WILL BE ACTIVE, 0 OTHERWISE'
      READ(5,*) IXE
C DETERMINE NUMBER OF DISCRETE VALUES
      WRITE(6,*), 'ENTER NUMBER OF GREY LEVELS'
      WRITE(6,*), '(NO MORE THAN',MAXDA,')'
      READ(5,*), NDA
C DETERMINE TEMPERATURE CONTROL PARAMETERS
      WRITE(6,*), 'ENTER STARTING TEMPERATURE, EVEN IF'
      WRITE(6,*), 'THIS IS FROM A PREVIOUS RUN'
      READ(5,*), TO
      WRITE(6,*), 'ENTER NUMBER OF ITERATIONS BEFORE INITIATION'
      WRITE(6,*), 'OF ANNEALING'
      READ(5,*), NO
C DETERMINE STARTING AND STOPPING ITERATIONS
      WRITE(6,*), 'ENTER NUMBER OF FIRST ITERATION FOR THIS RUN'
      READ(5,*), NSTART
      WRITE(6,*), 'ENTER NUMBER OF LAST ITERATION FOR THIS RUN'
      READ(5,*), NSTOP
C GET SEED FOR RANDOM NUMBER GENERATOR
      WRITE(6,*), 'ENTER SEED FOR RANDOM NUMBER GENERATOR'
      READ(5,*), SEED
C IF GOAL IS RESTORATION, AND THERE IS A DEGRADATION, THEN
C DETERMINE THE STANDARD ERROR OF ANY NOISE THAT HAS BEEN ADDED TO
C THE IMAGE AND COMPUTE THE TOTAL SIGMA SQUARED ("SIGSQD")
      IF (IS.EQ.0.AND.ID.EQ.1) THEN
         WRITE(6,*), 'ENTER STANDARD ERROR OF ADDED NOISE (0 IF NO'
         WRITE(6,*), 'NOISE HAS BEEN INTRODUCED)'
         READ(5,*), ADSIG
         SIGSQD=ADSIG**2+SIGMA**2
      ENDIF
C READ IN DATA
      IF (IP.EQ.1) THEN
         DO 1 J=1,NY
              READ(1,6) (XP(I,J),I=1,NX)
1        CONTINUE
         DO 3 K=1,2
         DO 4 J=1,NY
              READ(1,6) (XE(I,J,K),I=1,NX)
```

```
4        CONTINUE                                                       REA00780
3        CONTINUE                                                       REA00790
6        FORMAT(10F7.2)                                                 REA00800
         CLOSE(UNIT=1)                                                  REA00810
      ENDIF                                                             REA00820
      IF (ID.EQ.1) THEN                                                 REA00830
         DO 7 J=1,NY                                                    REA00840
              READ(2,6) (XPO(I,J),I=1,NX)                               REA00850
7        CONTINUE                                                       REA00860
         CLOSE(UNIT=2)                                                  REA00870
      ENDIF                                                             REA00880
      IF (IS.EQ.0.AND.ID.EQ.0.AND.IP.EQ.0) THEN                         REA00890
         DO 9 J=1,NY                                                    REA00900
              READ(3,6) (XP(I,J),I=1,NX)                                REA00910
9        CONTINUE                                                       REA00920
         CLOSE(UNIT=3)                                                  REA00930
      ENDIF                                                             REA00940
C INITIALIZE DATA ARRAYS.  ALL NONPIXEL STRUCTURES ARE                  REA00950
C INITIALIZED TO "NOT PRESENT", UNLESS THERE WAS                        REA00960
C PREVIOUS PROCESSING.                                                  REA00970
      IF (ID.EQ.1.AND.IP.EQ.0) THEN                                     REA00980
         DO 15 I=1,NX                                                   REA00990
         DO 20 J=1,NY                                                   REA01000
              XP(I,J)=XPO(I,J)                                          REA01010
20       CONTINUE                                                       REA01020
15       CONTINUE                                                       REA01030
      ENDIF                                                             REA01040
      IF (IS.EQ.1.AND.IP.EQ.0) THEN                                     REA01050
         DO 60 I=1,NX                                                   REA01060
         DO 70 J=1,NY                                                   REA01070
              XP(I,J)=PMIN+(PMAX-PMIN)*GGUBFS(SEED)                     REA01080
70       CONTINUE                                                       REA01090
60       CONTINUE                                                       REA01100
      ENDIF                                                             REA01110
      IF (IP.EQ.0) THEN                                                 REA01120
         DO 75 K=1,2                                                    REA01130
         DO 80 J=1,NY                                                   REA01140
         DO 90 I=1,NX                                                   REA01150
            XE(I,J,K)=0.0                                               REA01160
90       CONTINUE                                                       REA01170
80       CONTINUE                                                       REA01180
```

```
75        CONTINUE
      ENDIF
C INITIALIZE DUMMY BOUNDARIES
      DO 100 J=0,NY+1
         XP(0,J)=1000.0
         XP(NX+1,J)=1000.0
100   CONTINUE
      DO 110 I=1,NX
         XP(I,0)=1000.0
         XP(I,NY+1)=1000.0
110   CONTINUE
      DO 120 I=-1,NX+2
         XE(I,-1,1)=0.0
         XE(I,-1,2)=0.0
         XE(I,0,1)=0.0
         XE(I,0,2)=0.0
         XE(I,NY,2)=0.0
         XE(I,NY+1,1)=0.0
         XE(I,NY+1,2)=0.0
         XE(I,NY+2,1)=0.0
         XE(I,NY+2,2)=0.0
120   CONTINUE
      DO 130 J=-1,NY+2
         XE(-1,J,1)=0.0
         XE(-1,J,2)=0.0
         XE(0,J,1)=0.0
         XE(0,J,2)=0.0
         XE(NX,J,2)=0.0
         XE(NX+1,J,1)=0.0
         XE(NX+1,J,2)=0.0
         XE(NX+2,J,1)=0.0
         XE(NX+2,J,2)=0.0
130   CONTINUE
      END
```

4.4 Subroutine WRITEO.

The output routine WRITEO writes the output image file to the disk.

```
      SUBROUTINE WRITEO                                                 WRI00010
C SET UP DATA STRUCTURES                                                WRI00020
      INCLUDE (COMMON)                                                  WRI00030
C TYPES                                                                 WRI00040
      INTEGER I, J, K                                                   WRI00050
C WRITE OUTPUT TO UNIT 4                                                WRI00060
      DO 1 J=1,NY                                                       WRI00070
         WRITE(4,6) (XP(I,J),I=1,NX)                                    WRI00080
1     CONTINUE                                                          WRI00090
      DO 3 K=1,2                                                        WRI00100
      DO 4 J=1,NY                                                       WRI00110
         WRITE(4,6) (XE(I,J,K),I=1,NX)                                  WRI00120
4     CONTINUE                                                          WRI00130
3     CONTINUE                                                          WRI00140
6     FORMAT(10F7.2)                                                    WRI00150
      CLOSE(UNIT=4)                                                     WRI00160
      END                                                               WRI00170
```

4.5 Subroutine ITXP.

The subroutine ITXP guides the execution of the relaxation algorithm for the pixel process X^P.

```
      SUBROUTINE ITXP
C SET UP DATA STRUCTURES
      INCLUDE (COMMON)
C TYPES
      INTEGER I,J,K
      REAL EP(MAXDA), SUM(MAXDA), TOT, EMIN, EMAX, NRAND
C EXTERNAL FUNCTIONS CALLED
      REAL GGUBFS
C ITERATE PIXEL VALUES
      DO 10 J=1,NY
      DO 20 I=1,NX
C COMPUTE ENERGY VECTOR FOR PIXEL (I,J) AND STORE IN EP  EP(K)
C IS THE RELATIVE ENERGY FOR XP(I,J) AT THE K'TH DISCRETE VALUE
      CALL PIXEN(I,J,EP)
C PREVENT OVERFLOWS AND UNDERFLOWS BY RESCALING AND TRUNCATING EP
      EMIN =EP(1)
      DO 5 K=2,NDA
        IF (EP(K).LT.EMIN) THEN
           EMIN=EP(K)
        ENDIF
5     CONTINUE
      EMAX=T*20.0
      DO 6 K=1,NDA
         EP(K)=MIN(EMAX,EP(K)-EMIN)
6     CONTINUE
C UPDATE VALUE OF XP(I,J)
      SUM(1)=EXP(-EP(1)/T)
      DO 30 K=2,NDA
         SUM(K)=SUM(K-1)+EXP(-EP(K)/T)
30    CONTINUE
      NRAND=GGUBFS(SEED)
      TOT=SUM(NDA)
      DO 40 K=1,NDA
         IF (NRAND.LE.(SUM(K)/TOT)) THEN
              XP(I,J)=PMIN+((PMAX-PMIN)*(K-1))/(NDA-1)
              GO TO 20
         ENDIF
```

```
 40     CONTINUE                                                           ITX00380
 20     CONTINUE                                                           ITX00390
 10     CONTINUE                                                           ITX00400
        END                                                                ITX00410
```

4.6 Subroutine PIXEN.

The subroutine PIXEN is called by ITXP and returns the vector of (relative) energies that determine the local conditional distribution of the possible values for the pixel process at an arbitrary site.

```
C PIXEN(I,J,EP) COMPUTES THE RELATIVE ENERGY FOR THE NDA DIFFERENT
C POSSIBLE LEVELS OF PIXEL (I,J). THESE ARE RETURNED VIA EP(MAXDA).
      SUBROUTINE PIXEN(I,J,EP)
C SET UP DATA STRUCTURES
      INCLUDE (COMMON)
C TYPES
      INTEGER I, J, K
      REAL EP(MAXDA), ADIFF, XPTEMP, INC
C INITIALIZE EP
      DO 10 K=1,NDA
         EP(K)=0.0
10    CONTINUE
C COMPUTE DEGRADATION CONTRIBUTION TO ENERGY (IF ANY)
      IF (ID.EQ.1) THEN
         CALL PIXENO(I,J,EP)
      ENDIF
C COMPUTE PURE PIXEL CONTRIBUTION TO ENERGY
      INC=(PMAX-PMIN)/(NDA-1)
      DO 20 K=1,NDA
         XPTEMP=PMIN+INC*(K-1)
C PIXEL TO UPPER LEFT:
      IF ((XE(I-1,J,1)+XE(I-1,J-1,2))*
     M    (XE(I-1,J-1,1)+XE(I,J-1,2)).LT..5) THEN
         ADIFF=ABS((XPTEMP-XP(I-1,J-1))/CE1B)
         EP(K)=EP(K)-CE1A*DIAG/(1.0+ADIFF*ADIFF)
      ENDIF
C PIXEL ABOVE:
      IF (XE(I,J-1,2).LE..5) THEN
         ADIFF=ABS((XPTEMP-XP(I,J-1))/CE1B)
         EP(K)=EP(K)-CE1A/(1.0+ADIFF*ADIFF)
      ENDIF
C PIXEL TO UPPER RIGHT:
      IF ((XE(I,J-1,2)+XE(I,J-1,1))*
     M    (XE(I,J,1)+XE(I+1,J-1,2)).LT..5) THEN
         ADIFF=ABS((XPTEMP-XP(I+1,J-1))/CE1B)
         EP(K)=EP(K)-CE1A*DIAG/(1.0+ADIFF*ADIFF)
      ENDIF                                                             PIX00370
C PIXEL TO LEFT:                                                        PIX00380
      IF (XE(I-1,J,1).LE..5) THEN                                       PIX00390
         ADIFF=ABS((XPTEMP-XP(I-1,J))/CE1B)                             PIX00400
         EP(K)=EP(K)-CE1A/(1.0+ADIFF*ADIFF)                             PIX00410
      ENDIF                                                             PIX00420
C PIXEL TO RIGHT:                                                       PIX00430
      IF (XE(I,J,1).LE..5) THEN                                         PIX00440
         ADIFF=ABS((XPTEMP-XP(I+1,J))/CE1B)                             PIX00450
         EP(K)=EP(K)-CE1A/(1.0+ADIFF*ADIFF)                             PIX00460
      ENDIF                                                             PIX00470
C PIXEL TO LOWER LEFT:                                                  PIX00480
      IF ((XE(I-1,J,2)+XE(I-1,J,1))*                                    PIX00490
     M    (XE(I-1,J+1,1)+XE(I,J,2)).LT..5) THEN                         PIX00500
         ADIFF=ABS((XPTEMP-XP(I-1,J+1))/CE1B)                           PIX00510
         EP(K)=EP(K)-CE1A*DIAG/(1.0+ADIFF*ADIFF)                        PIX00520
      ENDIF                                                             PIX00530
C PIXEL BELOW:                                                          PIX00540
      IF (XE(I,J,2).LE..5) THEN                                         PIX00550
         ADIFF=ABS((XPTEMP-XP(I,J+1))/CE1B)                             PIX00560
         EP(K)=EP(K)-CE1A/(1.0+ADIFF*ADIFF)                             PIX00570
      ENDIF                                                             PIX00580
C PIXEL TO LOWER RIGHT:                                                 PIX00590
      IF ((XE(I,J,2)+XE(I,J+1,1))*                                      PIX00600
     M    (XE(I,J,1)+XE(I+1,J,2)).LT..5) THEN                           PIX00610
         ADIFF=ABS((XPTEMP-XP(I+1,J+1))/CE1B)                           PIX00620
         EP(K)=EP(K)-CE1A*DIAG/(1.0+ADIFF*ADIFF)                        PIX00630
      ENDIF                                                             PIX00640
20    CONTINUE                                                          PIX00650
      END                                                               PIX00660
```

4.7 Subroutine PIXENO.

The subroutine PIXENO is called by PIXEN and returns that part of the local energy vector attributable to the difference between the observed image and the current state of the restoration.

```
C PIXENO(I,J,EP) COMPUTES THE DEGRADATION CONTRIBUTION TO
C THE RELATIVE ENERGY FOR THE NDA DIFFERENT POSSIBLE LEVELS
C OF PIXEL (I,J).  THESE ARE RETURNED VIA EP(MAXDA).
      SUBROUTINE PIXENO(I,J,EP)
C SET UP DATA STRUCTURES
      INCLUDE (COMMON)
C TYPES
      INTEGER I, J, K
      REAL EP(MAXDA), XPTEMP, INC, TSIGSQ
C COMPUTE DEGREDATION CONTRIBUTION TO ENERGY
      INC=(PMAX-PMIN)/(NDA-1)
      TSIGSQ=2*SIGSQD
      DO 10 K=1,NDA
         XPTEMP=PMIN+INC*(K-1)
         EP(K)=EP(K)+(XPTEMP-XPO(I,J))**2/TSIGSQ
10    CONTINUE
      END
```

4.8 Subroutine ITXE.

The subroutine ITXE guides the execution of the relaxation algorithm for the edge process X^E.

```
      SUBROUTINE ITXE
C SET UP DATA STRUCTURES
      INCLUDE (COMMON)
C TYPES
      INTEGER I, J, K
      REAL PON,EXPO
C EXTERNAL FUNCTIONS CALLED
      REAL DEE
C ITERATE EDGE PROCESS
      DO 10 K=1,2
      DO 20 J=1,NY+1-K
      DO 30 I=1,NX-2+K
         EXPO=MIN(10.0,MAX(-10.0,DEE(I,J,K)/T))
         PON=1/(1+EXP(EXPO))
         IF (GGUBFS(SEED).LE.PON) THEN
              XE(I,J,K)=1.0
         ELSE
              XE(I,J,K)=0.0
         ENDIF
30    CONTINUE
20    CONTINUE
10    CONTINUE
      END
```

4.9 Subroutine DEE.

The subroutine DEE is called by ITXE and computes the energy difference between the states “on” and “off” for the edge element at an arbitrary edge site.

```
C DEE CALCULATES THE ENERGY DIFFERENCE (DELTA ENERGY) BETWEEN
C EDGE ELEMENT (I,J,K) IN STATE 1 (ON) AND EDGE ELEMENT (I,J,K)
C IN STATE 0 (OFF).
      REAL FUNCTION DEE(I,J,K)
C SET UP DATA STRUCTURES
      INCLUDE (COMMON)
C TYPES
      INTEGER I, J, K, NON
      REAL HOLD, RON, ADIFF
C INITIALIZE DEE
      DEE=0.0
C COMPUTE NONDIAGONAL PIXEL/EDGE CONTRIBUTION
      ADIFF=ABS((XP(I,J)-XP(I+2-K,J+K-1))/CE1B)
      DEE=DEE+CE1A/(1.0+ADIFF*ADIFF)
C COMPUTE NONDIAGONAL "BOND-BREAKING" PENALTY
      DEE=DEE-CE2B
C COMPUTE 4-CLIQUE TERMS, INCLUDING DIAGONAL PIXEL/EDGE
C TERMS AND DIAGONAL BOND-BREAKING TERMS
      HOLD=XE(I,J,K)
      XE(I,J,K)=1.0
      IF (K.EQ.1.AND.J.GT.1) THEN
         RON=XE(I,J-1,1)+XE(I+1,J-1,2)+XE(I,J,1)+XE(I,J-1,2)
         NON=NINT(RON)
         IF (NON.EQ.1) THEN
            DEE=DEE+3*CE2A
         ELSEIF (NON.EQ.2) THEN
            DEE=DEE-2*CE2A
            IF (XE(I,J-1,2).GT..5) THEN
               DEE=DEE-CE2B
               ADIFF=ABS((XP(I,J)-XP(I+1,J-1))/CE1B)
               DEE=DEE+CE1A*DIAG/(1.0+ADIFF*ADIFF)
            ELSEIF (XE(I+1,J-1,2).GT..5) THEN
               DEE=DEE-CE2B
               ADIFF=ABS((XP(I,J-1)-XP(I+1,J))/CE1B)
               DEE=DEE+CE1A*DIAG/(1.0+ADIFF*ADIFF)
            ELSE
               DEE=DEE-2*CE2B
               ADIFF=ABS((XP(I,J)-XP(I+1,J-1))/CE1B)                    DEE00380
               DEE=DEE+CE1A*DIAG/(1.0+ADIFF*ADIFF)                      DEE00390
               ADIFF=ABS((XP(I,J-1)-XP(I+1,J))/CE1B)                    DEE00400
               DEE=DEE+CE1A*DIAG/(1.0+ADIFF*ADIFF)                      DEE00410
            ENDIF                                                       DEE00420
         ELSEIF (NON.EQ.3) THEN                                         DEE00430
            DEE=DEE+CE2A                                                DEE00440
            IF (XE(I+1,J-1,2).LT..5) THEN                               DEE00450
               DEE=DEE-CE2B                                             DEE00460
               ADIFF=ABS((XP(I,J)-XP(I+1,J-1))/CE1B)                    DEE00470
               DEE=DEE+CE1A*DIAG/(1.0+ADIFF*ADIFF)                      DEE00480
            ELSEIF (XE(I,J-1,2).LT..5) THEN                             DEE00490
               DEE=DEE-CE2B                                             DEE00500
               ADIFF=ABS((XP(I,J-1)-XP(I+1,J))/CE1B)                    DEE00510
               DEE=DEE+CE1A*DIAG/(1.0+ADIFF*ADIFF)                      DEE00520
            ENDIF                                                       DEE00530
         ELSEIF (NON.EQ.4) THEN                                         DEE00540
            DEE=DEE+CE2A                                                DEE00550
         ENDIF                                                          DEE00560
      ENDIF                                                             DEE00570
      IF (K.EQ.1.AND.J.LT.NY) THEN                                      DEE00580
         RON=XE(I,J,1)+XE(I+1,J,2)+XE(I,J+1,1)+XE(I,J,2)                DEE00590
         NON=NINT(RON)                                                  DEE00600
         IF (NON.EQ.1) THEN                                             DEE00610
            DEE=DEE+3*CE2A                                              DEE00620
         ELSEIF (NON.EQ.2) THEN                                         DEE00630
            DEE=DEE-2*CE2A                                              DEE00640
            IF (XE(I,J,2).GT..5) THEN                                   DEE00650
               DEE=DEE-CE2B                                             DEE00660
               ADIFF=ABS((XP(I,J)-XP(I+1,J+1))/CE1B)                    DEE00670
               DEE=DEE+CE1A*DIAG/(1.0+ADIFF*ADIFF)                      DEE00680
            ELSEIF (XE(I+1,J,2).GT..5) THEN                             DEE00690
               DEE=DEE-CE2B                                             DEE00700
               ADIFF=ABS((XP(I,J+1)-XP(I+1,J))/CE1B)                    DEE00710
               DEE=DEE+CE1A*DIAG/(1.0+ADIFF*ADIFF)                      DEE00720
            ELSE                                                        DEE00730
               DEE=DEE-2*CE2B                                           DEE00740
               ADIFF=ABS((XP(I,J)-XP(I+1,J+1))/CE1B)                    DEE00750
               DEE=DEE+CE1A*DIAG/(1.0+ADIFF*ADIFF)                      DEE00760
               ADIFF=ABS((XP(I,J+1)-XP(I+1,J))/CE1B)                    DEE00770
               DEE=DEE+CE1A*DIAG/(1.0+ADIFF*ADIFF)                      DEE00780
            ENDIF                                                       DEE00790
         ELSEIF (NON.EQ.3) THEN                                         DEE00800
            DEE=DEE+CE2A                                                DEE00810
            IF (XE(I+1,J,2).LT..5) THEN                                 DEE00820
               DEE=DEE-CE2B                                             DEE00830
               ADIFF=ABS((XP(I,J)-XP(I+1,J+1))/CE1B)                    DEE00840
               DEE=DEE+CE1A*DIAG/(1.0+ADIFF*ADIFF)                      DEE00850
            ELSEIF (XE(I,J,2).LT..5) THEN                               DEE00860
               DEE=DEE-CE2B                                             DEE00870
               ADIFF=ABS((XP(I,J+1)-XP(I+1,J))/CE1B)                    DEE00880
               DEE=DEE+CE1A*DIAG/(1.0+ADIFF*ADIFF)                      DEE00890
            ENDIF                                                       DEE00900
         ELSEIF (NON.EQ.4) THEN                                         DEE00910
            DEE=DEE+CE2A                                                DEE00920
         ENDIF                                                          DEE00930
      ENDIF                                                             DEE00940
      IF (K.EQ.2.AND.I.GT.1) THEN                                       DEE00950
         RON=XE(I-1,J,1)+XE(I,J,2)+XE(I-1,J+1,1)+XE(I-1,J,2)            DEE00960
         NON=NINT(RON)                                                  DEE00970
         IF (NON.EQ.1) THEN                                             DEE00980
            DEE=DEE+3*CE2A                                              DEE00990
         ELSEIF (NON.EQ.2) THEN                                         DEE01000
            DEE=DEE-2*CE2A                                              DEE01010
            IF (XE(I-1,J,1).GT..5) THEN                                 DEE01020
               DEE=DEE-CE2B                                             DEE01030
               ADIFF=ABS((XP(I,J)-XP(I-1,J+1))/CE1B)                    DEE01040
               DEE=DEE+CE1A*DIAG/(1.0+ADIFF*ADIFF)                      DEE01050
            ELSEIF (XE(I-1,J+1,1).GT..5) THEN                           DEE01060
               DEE=DEE-CE2B                                             DEE01070
               ADIFF=ABS((XP(I-1,J)-XP(I,J+1))/CE1B)                    DEE01080
               DEE=DEE+CE1A*DIAG/(1.0+ADIFF*ADIFF)                      DEE01090
            ELSE                                                        DEE01100
               DEE=DEE-2*CE2B                                           DEE01110
               ADIFF=ABS((XP(I,J)-XP(I-1,J+1))/CE1B)                    DEE01120
               DEE=DEE+CE1A*DIAG/(1.0+ADIFF*ADIFF)                      DEE01130
               ADIFF=ABS((XP(I-1,J)-XP(I,J+1))/CE1B)                    DEE01140
               DEE=DEE+CE1A*DIAG/(1.0+ADIFF*ADIFF)                      DEE01150
            ENDIF                                                       DEE01160
         ELSEIF (NON.EQ.3) THEN                                         DEE01170
            DEE=DEE+CE2A                                                DEE01180
            IF (XE(I-1,J+1,1).LT..5) THEN                               DEE01190
```

```
               DEE=DEE-CE2B
               ADIFF=ABS((XP(I,J)-XP(I-1,J+1))/CE1B)
               DEE=DEE+CE1A*DIAG/(1.0+ADIFF*ADIFF)
            ELSEIF (XE(I-1,J,1).LT..5) THEN
               DEE=DEE-CE2B
               ADIFF=ABS((XP(I-1,J)-XP(I,J+1))/CE1B)
               DEE=DEE+CE1A*DIAG/(1.0+ADIFF*ADIFF)
            ENDIF
         ELSEIF (NON.EQ.4) THEN
            DEE=DEE+CE2A
         ENDIF
      ENDIF
      IF (K.EQ.2.AND.I.LT.NX) THEN
         RON=XE(I,J,1)+XE(I+1,J,2)+XE(I,J+1,1)+XE(I,J,2)
         NON=NINT(RON)
         IF (NON.EQ.1) THEN
            DEE=DEE+3*CE2A
         ELSEIF (NON.EQ.2) THEN
            DEE=DEE-2*CE2A
            IF (XE(I,J,1).GT..5) THEN
               DEE=DEE-CE2B
               ADIFF=ABS((XP(I,J)-XP(I+1,J+1))/CE1B)
               DEE=DEE+CE1A*DIAG/(1.0+ADIFF*ADIFF)
            ELSEIF (XE(I,J+1,1).GT..5) THEN
               DEE=DEE-CE2B
               ADIFF=ABS((XP(I,J+1)-XP(I+1,J))/CE1B)
               DEE=DEE+CE1A*DIAG/(1.0+ADIFF*ADIFF)
            ELSE
               DEE=DEE-2*CE2B
               ADIFF=ABS((XP(I,J)-XP(I+1,J+1))/CE1B)
               DEE=DEE+CE1A*DIAG/(1.0+ADIFF*ADIFF)
               ADIFF=ABS((XP(I,J+1)-XP(I+1,J))/CE1B)
               DEE=DEE+CE1A*DIAG/(1.0+ADIFF*ADIFF)
            ENDIF
         ELSEIF (NON.EQ.3) THEN
            DEE=DEE+CE2A
            IF (XE(I,J+1,1).LT..5) THEN
               DEE=DEE-CE2B
               ADIFF=ABS((XP(I,J)-XP(I+1,J+1))/CE1B)
               DEE=DEE+CE1A*DIAG/(1.0+ADIFF*ADIFF)
            ELSEIF (XE(I,J,1).LT..5) THEN
```

```
               DEE=DEE-CE2B                                             DEE01610
               ADIFF=ABS((XP(I,J+1)-XP(I+1,J))/CE1B)                    DEE01620
               DEE=DEE+CE1A*DIAG/(1.0+ADIFF*ADIFF)                      DEE01630
            ENDIF                                                       DEE01640
         ELSEIF (NON.EQ.4) THEN                                         DEE01650
            DEE=DEE+CE2A                                                DEE01660
         ENDIF                                                          DEE01670
      ENDIF                                                             DEE01680
C CONTRIBUTION FORM INHIBITION OF PARALLEL LINES                        DEE01690
      IF (K.EQ.1) THEN                                                  DEE01700
         DEE=DEE+CE2C*(XE(I-2,J,1)+XE(I-1,J,1)+XE(I+1,J,1)+XE(I+2,J,1)) DEE01710
      ELSE                                                              DEE01720
         DEE=DEE+CE2C*(XE(I,J-2,2)+XE(I,J-1,2)+XE(I,J+1,2)+XE(I,J+2,2)) DEE01730
      ENDIF                                                             DEE01740
      XE(I,J,K)=HOLD                                                    DEE01750
      END                                                               DEE01760
```

4.10 Function Subprogram GGUBFS.

The function subprogram GGUBFS is from the proprietary IMSL Library and is used to generate pseudorandom numbers that are independent and uniformly distributed on (0, 1). *The listing below should not be reproduced nor incorporated in any programs other than the present one unless its use is licensed on the system on which such a program is being developed.*

```
C   IMSL ROUTINE NAME   - GGUBFS
C
C-----------------------------------------------------------------------
C
C   COMPUTER            - IBM/SINGLE
C
C   LATEST REVISION     - JUNE 1, 1980
C
C   PURPOSE             - BASIC UNIFORM (0,1) RANDOM NUMBER GENERATOR -
C                           FUNCTION FORM OF GGUBS
C
C   USAGE               - FUNCTION GGUBFS (DSEED)
C
C   ARGUMENTS    GGUBFS - RESULTANT DEVIATE.
C                DSEED  - INPUT/OUTPUT DOUBLE PRECISION VARIABLE
C                           ASSIGNED AN INTEGER VALUE IN THE
C                           EXCLUSIVE RANGE (1.DO, 2147483647.DO).
C                           DSEED IS REPLACED BY A NEW VALUE TO BE
C                           USED IN A SUBSEQUENT CALL.
C
C   PRECISION/HARDWARE  - SINGLE/ALL
C
C   REQD. IMSL ROUTINES - NONE REQUIRED
C
C   NOTATION            - INFORMATION ON SPECIAL NOTATION AND
C                           CONVENTIONS IS AVAILABLE IN THE MANUAL
C                           INTRODUCTION OR THROUGH IMSL ROUTINE UHELP
C
C   COPYRIGHT           - 1978 BY IMSL, INC. ALL RIGHTS RESERVED.
C
C   WARRANTY            - IMSL WARRANTS ONLY THAT IMSL TESTING HAS BEEN
C                           APPLIED TO THIS CODE. NO OTHER WARRANTY,
C                           EXPRESSED OR IMPLIED, IS APPLICABLE.
C
```

```
C-----------------------------------------------------------------------GGU00350
C                                                                       GGU00360
      REAL FUNCTION GGUBFS (DSEED)                                      GGU00370
C                                  SPECIFICATIONS FOR ARGUMENTS         GGU00380
      DOUBLE PRECISION   DSEED                                          GGU00390
C                                  SPECIFICATIONS FOR LOCAL VARIABLES   GGU00400
      DOUBLE PRECISION   D2P31M,D2P31                                   GGU00410
C                                  D2P31M=(2**31) - 1                   GGU00420
C                                  D2P31 =(2**31)(OR AN ADJUSTED VALUE) GGU00430
      DATA               D2P31M/2147483647.DO/                          GGU00440
      DATA               D2P31 /2147483648.DO/                          GGU00450
C                                  FIRST EXECUTABLE STATEMENT           GGU00460
      DSEED = DMOD(16807.DO*DSEED,D2P31M)                               GGU00470
      GGUBFS = DSEED / D2P31                                            GGU00480
      RETURN                                                            GGU00490
      END                                                               GGU00500
```

5 FLIR EXAMPLES.

The algorithm described in Sections 2 and 3 and implemented by the program of Section 4 has been applied to a variety of FLIR images provided by the Advanced Modeling Team at NV&EOL. The results of selected experiments are included here.

For these experiments, the model parameters were set on the basis of inspection of the digitized FLIR images to determine attributes such as dynamic range and noise-variance and on the basis of the insights and interpretations of the model parameters described in Section 3.

In each of the photographs in Appendix B, the upper-left panel contains a 32 × 32 section of the observed image. The upper-right panel contains the result of fifty iterations of the stochastic relaxation algorithm, with annealing. The lower-left panel contains the original observed image ***plus additional noise*** having standard deviation 8. The lower-right panel contains the result of fifty iterations of the stochastic relaxation algorithm, with annealing, applied to the noise corrupted image.

The model and program parameters for the experiments are given in the following table:

Model	Program	Value
θ_1	CE1A	10.4
B	CE1B	4.0
θ_2	CE2B	1.66
θ_3	CE2A	1.0
ξ_1		-4
ξ_2		-3
ξ_3		-2
ξ_4		-1
ξ_5		-1
ξ_6		0
	PMIN	40
	PMAX	238
	MAXDA	100
	NDA	100

For the original observed images, the standard deviation SIGMA of the additive noise presumed to be degrading the ideal image was set to 5.

For the images to which noise was added, the standard deviation in the restoration algorithm was set to $\sqrt{25+64} = 9.43$.

Eight figures are included in Appendix B.

6 REFERENCES.

1. V. Cerný, "A thermodynamic approach to the travelling salesman problem: an efficient simulation algorithm," Inst. Phys. & Biophys., Comenius Univ., Bratislava. 1982 (preprint).

2. D. Geman and S. Geman, "Bayesian image analysis," NATO ASI Series. Vol. F20. *Disordered Systems and Biological Organization*, Springer-Verlag, Berlin. 1986.

3. D. Geman, S. Geman and C. Graffigne, "Locating texture and object boundaries," NATO ASI Series, P. Devijver (editor), Springer-Verlag, Heidelberg, 1986.

4. S. Geman and D. Geman, "Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images," IEEE Trans. Pattern Anal. Machine Intell., **6**, 721-741, 1984.

5. S. Geman and D.E. McClure, "Bayesian image analysis: an application to single photon emission tomography," *1985 Proceedings of the American Statistical Association, Statistical Computing Section,* 1985.

6. B. Gidas, "Non-stationary Markov chains and convergence of the annealing algorithm," J. Stat. Phys., **39**, 73-131, 1985.

7. B. Gidas, "A renormalization group approach to image processing problems," Division of Applied Mathematics, Brown University, 1986 (preprint).

8. B. Gidas, Personal communication, 1985.

9. U. Grenander, "Tutorial in pattern theory," Division of Applied Mathematics, Brown University, 1983 (preprint).

10. B. Hajek, "Cooling schedules for optimal annealing," Mathematics of Operations Research, 1985.

11. A.R. Hansen and E.M. Riseman, "Segmentation of natural scenes," *Computer Vision Systems,* Academic Press, New York, 1978.

12. S. Kirkpatrick, C.D. Gellatt and M.P. Vecchi, "Optimization by simulated annealing," Science, **220**, 671-680, 1983.

13. J. Marroquin, S. Mitter and T. Poggio, "Probabilistic solution of ill-posed problems in computational vision," Artificial Intelligence Lab. Technical Report, MIT, 1985.

14. N. Metropolis, A.W. Rosenbluth, M.N. Rosenbluth, A.H. Teller and E. Teller, "Equations of state calculations by fast computing machines," J. Chem. Phys., **21**, 1087-1091, 1953.

15. V. Mirelli, "IR sensor model," Internal working memorandum, Advanced Modeling Team, Night Vision & Electro-Optics Laboratory, Ft. Belvoir, 1984.

16. J. Ratches et. al., "Night Vision Laboratory static performance model for thermal viewing systems," Research and Development Technical Report ECOM-7043, U.S. Army Electronics Command, Ft. Monmouth, 1975.

17. B. Saleh, *Photoelectron Statistics*, Springer-Verlag, 1978.

A COMPLEX SYSTEMS WORKING PAPERS.

During the course of the modeling project, a number of internal working papers were prepared describing progress and research plans for specific aspects of the research effort. These papers were not intended for general distribution. Nonetheless, because of the direct cooperation with the Advanced Modeling Team at NV&EOL, the working papers were all shared with the leaders of the team. Titles of the working papers directly related to the image analysis problems at NV&EOL include:

- An entropy approach to relaxation time, April 1983.
- Updating schemes for image processing, June 1983.
- Parameter estimation for some Markov random fields, August 1983.
- Synthesis of partition patterns, August 1983.
- Synthesis of surface patterns, August 1983.
- A computer experiment with sweep areas, October 1983.
- Some experiments with partition, shape, and network patterns, October 1983.
- Simulating cold patterns is difficult, November 1983.
- Stochastic relaxation for some continuous generator spaces, November 1983.
- Remarks on annealing schedules, December 1983.
- Recognizing objects, March 1984.
- Non-localized generators, May 1984.
- Parameter estimation for Markov random fields with hidden variables and experiments with the EM algorithm, August 1984.
- Aspects of image processing, September 1984.
- Software for image processing experiments, November 1984.
- Preliminaries to target identification in IR-pictures, April 1985.
- Recognizing patterns in the presence of nuisance parameters, February 1986.

- Modeling and recognition of textures, March 1986.
- Parallel logic under uncertainty, continued and applied to the car experiment, August 1986.

B. FIGURES

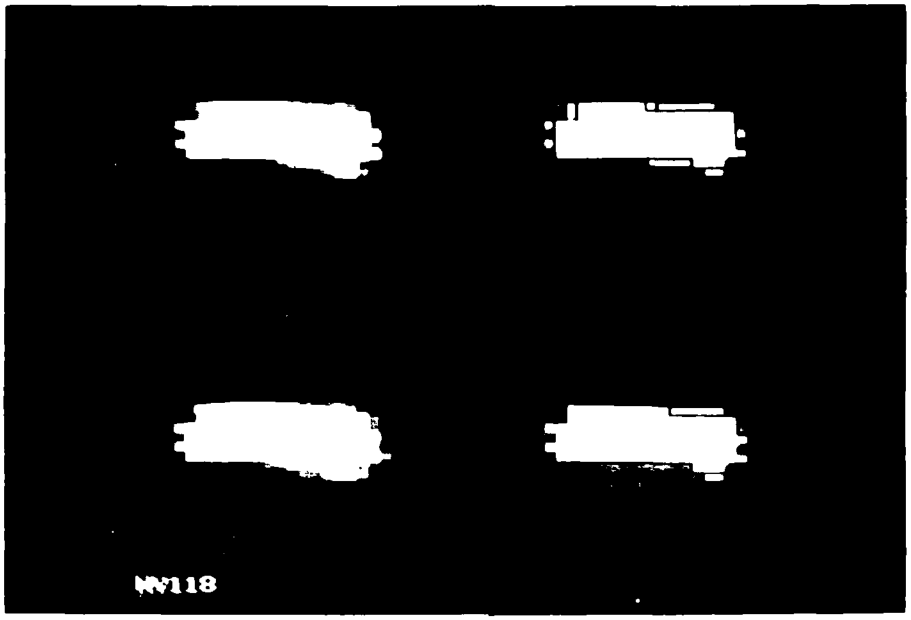


FIGURE 1

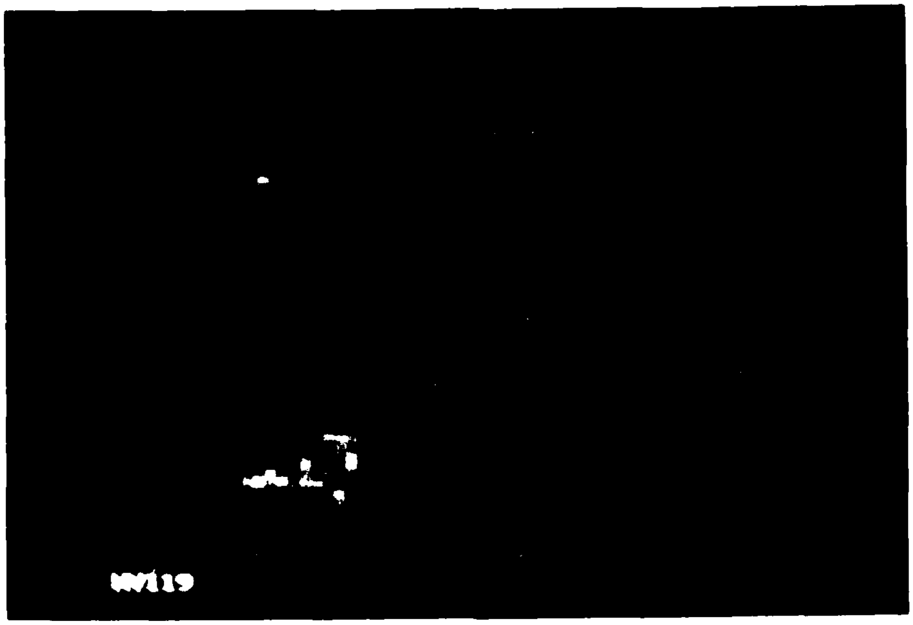


FIGURE 2

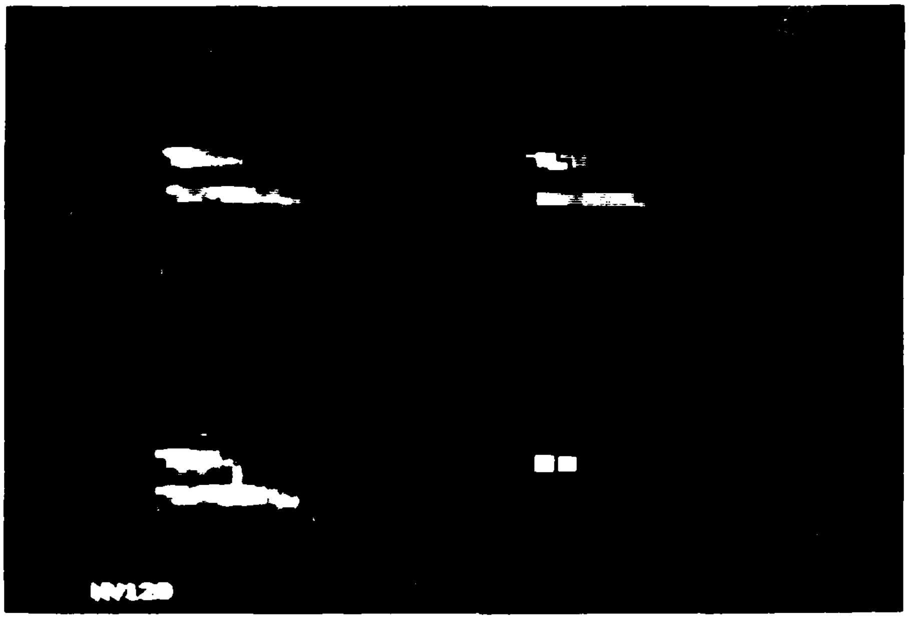


FIGURE 3

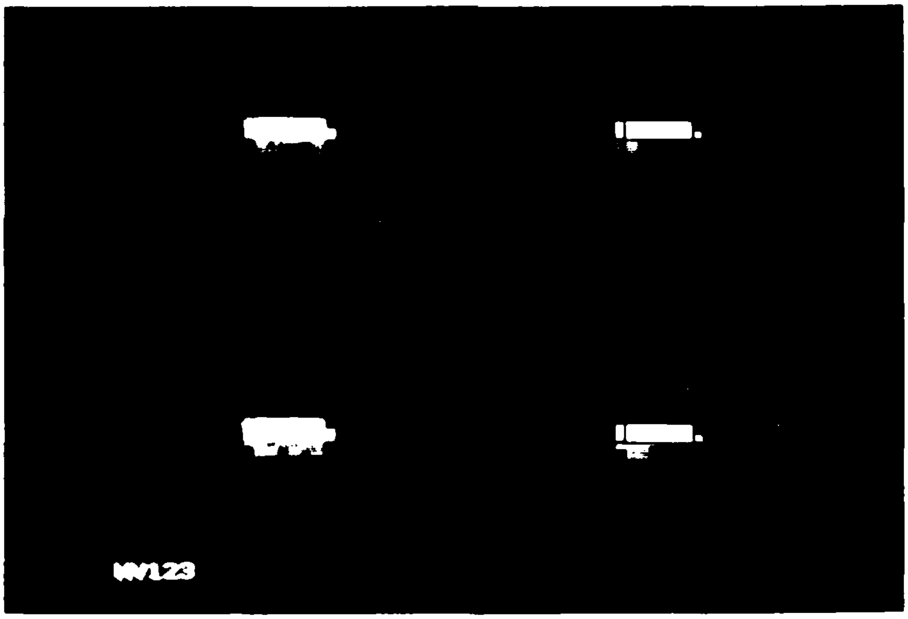


FIGURE 4

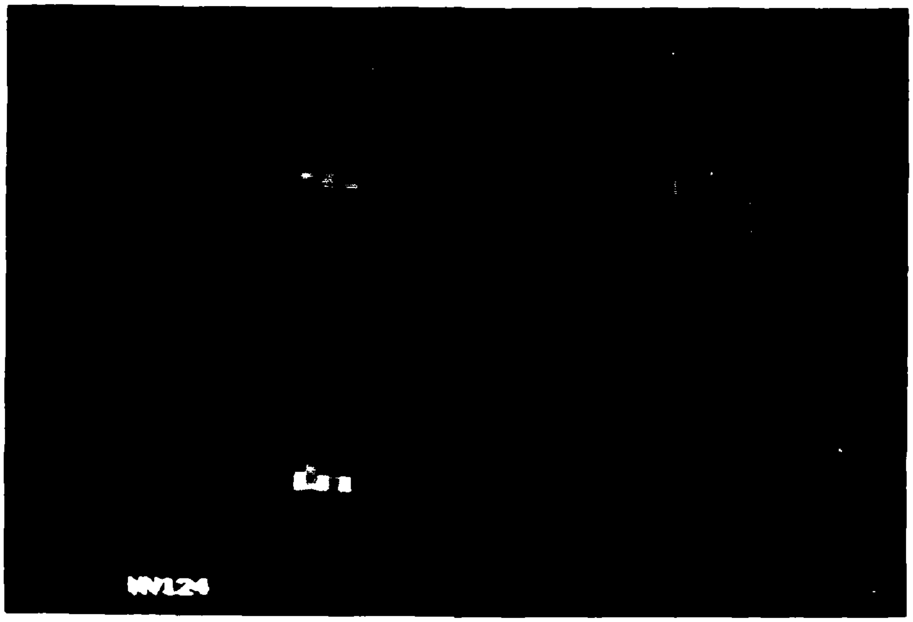


FIGURE 5

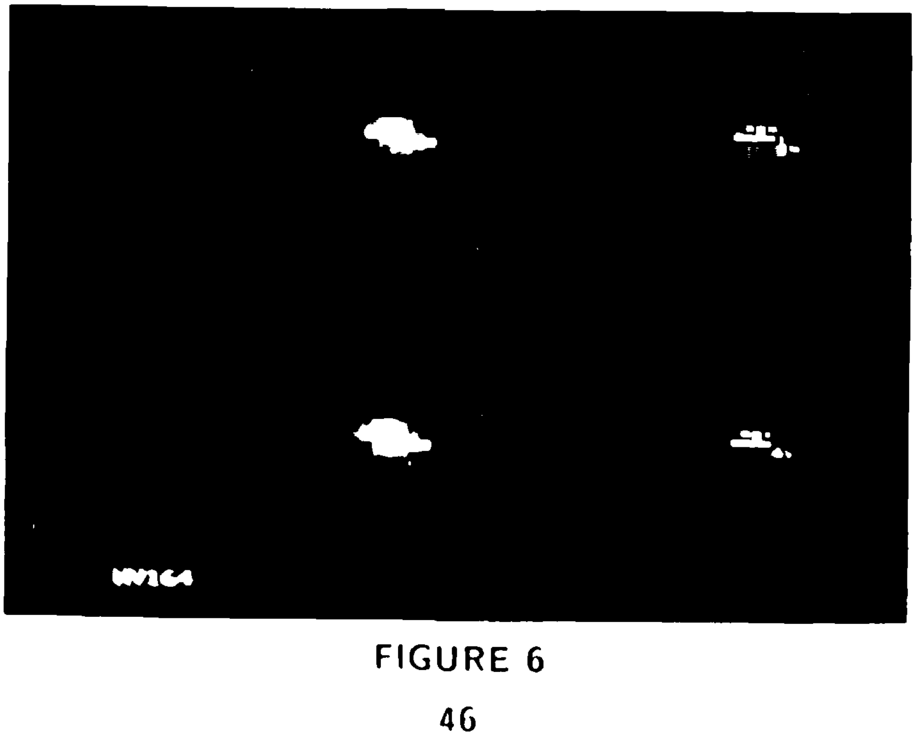


FIGURE 6

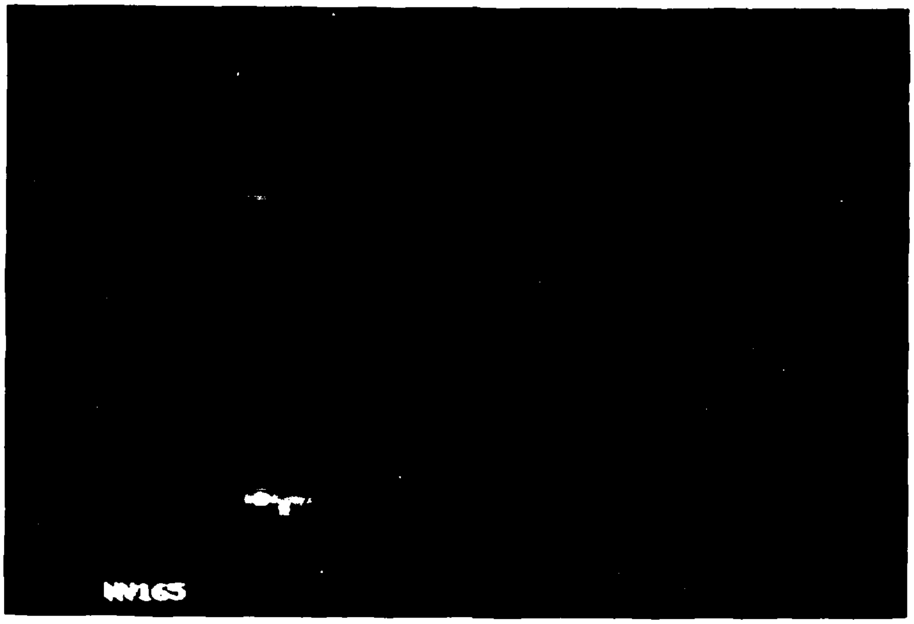


FIGURE 7

FIGURE 8

END

5-87

DTIC